Behavioral Heterogeneity, Cooperation, and Collective Construction

Geoff Nitschke
Ikegami Laboratory, Interdisciplinary Studies Department
University of Tokyo, Japan
Email: geoff@sacral.c.u-tokyo.ac.jp

***Abstract*— This paper evaluates two *Neuro-Evolution* (NE) methods to adapt controllers in simulated robot teams. The first method evolves controllers with fixed topologies and adapts team size as a function of task complexity. The second method evolves controller topology as a function of task complexity, but keeps team sizes constant. These methods are: *Collective Neuro-Evolution 2* (CONE-2), and *Neuro-Evolution for Augmenting Topologies* (NEAT). CONE-2 and NEAT are comparatively tested in a collective construction task. The goal is to ascertain the most appropriate controller evolution method for adapting teams to solve a collective construction task, with varying cooperative behavior requirements. Results indicate that CONE-2 is most effective at adapting controllers as the complexity of the task increases. In environments where multiple forms of cooperative behavior are required, CONE-2 evolves teams with a higher average task performance. CONE-2 is demonstrated as being effective at evolving behavioral heterogeneity in teams, which results in a higher team fitness, comparative to NEAT evolved teams, in environments that require cooperation.**

I. Introduction

In fields of research such as multi-robot systems [20], one objective is to replicate desirable collective behaviors exhibited in biological systems, and the underlying mechanisms responsible for such collective behaviors. Such underlying mechanisms include emergent *behavioral specialization* and *cooperation*. *Collective behavior* refers to group behaviors that emerge from the local interaction of many individuals. For example, group behaviors in social insect colonies [2], multi-cellular organisms [9], and economies of a nation [18].

Previous research on the adaptation of simulated robot team behaviors has demonstrated that the interactions of complementary specialized behaviors results in the formation of effective collective behaviors (for example, multi-robot coordination and cooperation), which in turn increases team task performance [17], [11] [14], [15].

In the study of controller evolution methods, the most appropriate adaptation method for a given task and environment is often unclear. This study applies *CONE-2*, an extension of *Collective Neuro-Evolution* [13] (CONE), using *Neuro-Evolution* (NE) [26] to evolve simulated robot teams to solve a collective construction task. CONE is a NE [6] controller design method that uses cooperative co-evolution [16] to adapt teams of *Artificial Neural Network* (ANN) controllers to solve collective behavior tasks. The key contribution of CONE is its use of *emergent behavioral specialization* as a problem solving mechanism. This allows CONE to increase task performance or attain collective behavior solutions that could not otherwise be attained (without specialization).

The key contribution of CONE-2 is its adaptation of team size (in this study, the number of robots) as a function of task complexity (in this study, a collective construction task). CONE-2 is comparatively tested with the *Neuro-Evolution of Augmenting Topologies* (NEAT) method [23]. Where as, CONE-2 uses cooperative co-evolution to adapt teams of ANN controllers, NEAT uses competitive co-evolution [19]. CONE-2 uses controllers with fixed topologies (evolving connection weights) and adapts team size, where as NEAT adapts controller topology (as a function of task complexity), and keeps team size fixed. Both NEAT and CONE (with which CONE-2 shares many similarities) have been applied to various complex, continuous, and noisy tasks [24], [25], [22], [14], [15]. However, CONE is most appropriate for solving collective behavior tasks that require controllers to adopt complementary specialized behaviors (that is, behaviorally heterogenous teams). CONE-2 and NEAT efficacy for evolving teams is tested in a range of collective construction task environments, where behavioral specialization is beneficial and varying degrees of cooperation are required.

The objective of this study is to gain an initial insight into the adaptive method types that are advantageous for controller evolution in teams operating in environments where *behavioral specialization* and *cooperation* are beneficial. Both behavioral specialization and cooperation are included as design aspects of this study's experiments, since previous work has demonstrated that the interactions of multiple complementary behavioral specializations facilitate emergent cooperative team behaviors [17], [11], [14], [15].

A. Research Goal, Hypothesis, and Task

The research goal is to demonstrate that CONE-2 evolves teams with *behavioral heterogeneity* that solve the collective construction task. Behavioral heterogeneity is defined as different robots in a team adopting complementary behavioral specializations. Previous research indicated that behavioral heterogeneity emerges in response to task and environment constraints mandating specialization [11].

The research hypothesis is that such behavioral heterogeneity facilitates cooperation, which results in higher team fitness (compared to NEAT evolved teams) in the collective construction task. This hypothesis was formulated given previous research results [14], [15].

This task requires a simulated robot team to gather blocks and cooperatively build a structure from gathered blocks in a *construction zone*. The complexity of this task is equated

with the degree of cooperation (number of robots required) to connect one block to another in the construction zone. Blocks are of types *A*, *B*, or *C*, where *construction rules* dictate how different block types connect together in a *construction sequence*. Task performance (team fitness) is the number of blocks correctly connected during a team's lifetime.

II. Collective Neuro-Evolution (Version 2)

Collective Neuro-Evolution - version 2 (CONE-2) is a cooperative co-evolution NE method, that extends CONE [13]. CONE-2's contribution is that it increases the number of genotype populations (from which controllers are evolved) as a function of task complexity. Thus, CONE-2 increases the number of controllers until a team size appropriate for task accomplishment is found. One controller (fully connected feed-forward ANN with a fixed topology) is evolved from each population. CONE-2 evolves hidden layer neuron connection weights from multiple sub-populations and combines these neurons into complete controllers.

CONE-2 starts with one population (P_0), segregated into *u* sub-populations, from which *u* hidden layer neurons are evolved. Each genotype in each sub-population encodes the connection weights of one hidden layer neuron. Each gene in each genotype is initialized to a random value in a given range. The single population (P_0) controller evolution process is the same as *Enforced Sub-Populations* (ESP) [8].

As CONE-2 initializes and adds new populations, *N* (where, $N \geq 2$) controllers are cooperatively co-adapted based on how well controllers cooperatively solve a task. An example of CONE-2 using three controllers (populations), and creating a new population is presented in figure 1 (left). CONE-2's process for constructing and evaluating *N* controllers evolved from *N* populations, is the same as used for Multi-Agent ESP [27] and CONE [13]. CONE-2 uses the following heuristics to adapt team size.

1) *Adding Population P_{n+1} ($n \geq 0$):* If fitness of the current team $\{ANN_0 \dots ANN_n\}$, evolved from populations $\{P_0 \dots P_n\}$ has not increased in *Z* generations (table III), a new population P_{n+1} is created. A new controller ANN_{n+1} is then evolved from P_{n+1}.
2) *Initializing Population P_{n+1}:* P_{n+1} is created with *u* sub-populations. Each sub-population is initialized with *o* genotypes (hidden layer neurons). For each new population: P_{n+1}, *u* and *o* are the same as for P_n. In order that P_{n+1} is able to evolve a controller from a beneficial part of the solution space, the genotypes of P_{n+1} are initialized based on one of the existing populations $\{P_0 \dots P_n\}$ (selected randomly). Each of the genotypes in P_{n+1} is initialized with the genotypes of an existing population. However, burst mutation with a Cauchy distribution [8] is applied to each gene of each genotype in P_{n+1} with a given probability. This P_{n+1} initialization procedure ensures that the new controller ANN_{n+1} is not be too dissimilar to the current team of *N* controllers. Also, the time taken for ANN_{n+1} to adapt to a beneficial behavioral role in the team is minimized since its behavior is based on an already functional controller behavior.

A. CONE-2 Genetic Operators

Unlike CONE (using inter-population recombination), in CONE-2 recombination only occurs within populations.

- *Recombination:* After all genotypes (in all populations) have been assigned a fitness and ranked, genotypes are recombined. For a given sub-population (in a given population), each genotype in the sub-population's *elite portion* (table III) is systematically selected and paired with another genotype (randomly selected from the elite portion). One-point crossover [4] is applied to each of these parent pairs. Enough child genotypes are produced to completely replace the current sub-population. This recombination and replacement procedure is repeated for every sub-population of every population.
- *Mutation:* After recombination, burst mutation [8] is applied to each genotype's gene with a given probability.

III. NEAT: Neuro-Evolution of Augmented Topologies

NEAT is a competitive co-evolution NE method that uses mechanisms for *historical gene marking*, *speciation*, and *complexification* [23], [24], [21] in its adaptation process.

Complexification is the incremental growth from minimal ANN controller topology. NEAT begins with a homogenous population of simple controllers (with no hidden nodes) and adapts connection weights and topology as a function of task complexity. Thus, NEAT biases the search towards minimal dimensional spaces and only increases search space dimensionality (adding controller structure) if the task requires it.

Speciation in NEAT calculates if two controllers will be in the same or a new species (according to a genotype compatibility threshold) after controllers have been recombined and mutated every generation. Speciating the population means controllers will only compete within their given species. This protects new innovations in controller topology adaptation.

Historical gene markings allow NEAT to add new structure and recombine controllers with differing topologies, since gene markings are evidence of controller homology.

NEAT has been successfully applied to various control tasks including double pole balancing [23], automobile control [22], and playing Go [24]. However, with the exception of NEAT extensions such as HyperNEAT, applied to certain multi-agent tasks [3], NEAT has not been applied to collective behavior tasks. In this study's collective construction task, NEAT uses homogenous fixed sized robot teams. That is, the current fittest controller is used for each robot.

A. NEAT: Genetic Operators

- *Mutation:* NEAT mutation adapts both connection weights and controller topology. To adapt controller weights, this application of NEAT uses burst mutation [8] to change each gene in each genotype with a given probability. NEAT controller topology adaptation is the basis of complexification and works via adding

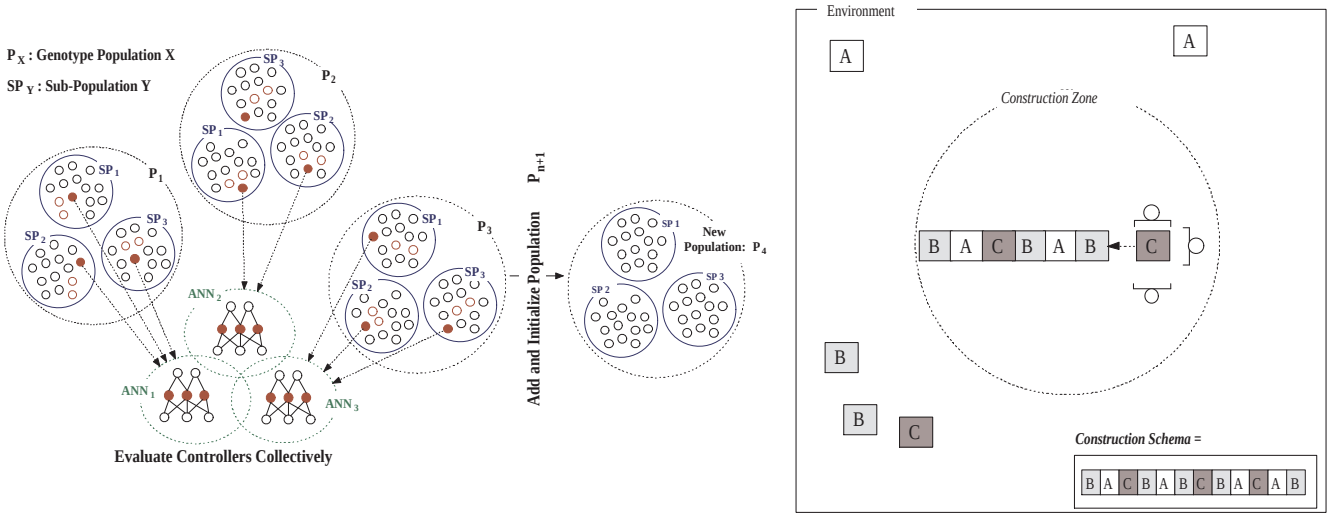

Fig. 1. *CONE-2 Example (Left):* CONE-2 adds population (P_n) / controllers (ANN_n) as a function of task complexity. Here, P_4 is initialized using mutated genotypes from P_3. *Collective Construction Task Example (Right):* Six of 12 blocks have been connected in the correct sequence (defined by the construction schema). Three robots are cooperatively connecting a type C block to the partially built object in the construction zone (not to scale).

genes to genotypes. New genes are new connections or controller nodes represented by a mutated genotype. These mutations are simultaneously applied to each genotype with a given probability. Added connection weights connect two previously unconnected nodes in a mutated controller. When a new node *c* is added, the existing connection between two existing nodes *a* and *b* is disabled. A new connection between nodes *a* and *c* (weight value = 1) is initialized. A second connection is initialized between node *c* and *b*, with the same weight value that previously connected nodes *a* and *b*. When a new gene is added, the new gene is assigned an incremented *global innovation number*.

- *Recombination:* Since NEAT tracks the historical origin of all genes using innovation numbers, this means that only homologous genes in two given controllers will be recombined. That is, NEAT only recombines genotypes (controllers) with ancestral genes in common. Matching genes are randomly selected for child genotypes. Disjoint and excess genes are inherited from the fitter parent, or at random in the case of equal fitness.

IV. Collective Construction Task and Experiments

Experiments test *n* robots in a bounded two dimensional continuous environment containing a distribution of type *A*, *B*, and *C* blocks. For CONE-2, $n = 1$ (initially), with a maximum of $n = 10$, and for NEAT, $n = 10$ (table III). The environment also contains a *construction zone*, where *N* gathered blocks are delivered and connected into a single object. Figure 1 (right) illustrates an example environment, containing three robots and 12 blocks. How different block types are connected is dictated by *construction rules* (table II), and the sequence in which different block types must be connected is defined by a *construction schema* (table I).

The construction rules regulate the difficulty of the construction process via requiring varying degrees of cooperation to make specific block connections. For example, the construction schema in figure 1 (right) mandates that the middle type C block be connected to a type B on either side. Making these connections requires two robots to cooperate (table II). In the collective construction task, cooperation refers to at least two robots simultaneously gripping and pushing a block to another block, to which it must connect. Cooperation is not required for robots to gather blocks. Thus, blocks are delivered to the construction zone by individual robots, but cooperation is often required for construction.

Previous research indicates that block type distributions affect emergent specialization during controller evolution [15], [12]. For example, in environment 9 (table I), block type *B* is particularly plentiful and block type *C* scarce. An effective collective construction behavior would be for most robots to grip type *B* blocks (after transporting them to the construction zone), and for few to grip type *C* blocks. Such a collective behavior would help minimize the time taken to connect all blocks in the construction zone. Related work [15], [12] indicated that such behavioral specialization is effective for increasing team task performance in collective construction tasks such as this.

The construction schemas (table I) were selected to address this study's research goal and hypothesis (section I-A). That is, to investigate the efficacy of CONE-2 versus NEAT for evolving behavioral specialization and cooperation that results from specialization. For example, environment 1 (table I) is not intended to encourage specialization, and no cooperation is needed to connect block types. However, for environments [2, 10], progressively more cooperation is needed to complete construction. For example, in environment 2, at the end of the construction process two robots are required to cooperate (to connect type *C* and *B* blocks). Where as, in environment 10, three robots must cooperate to

TABLE I

BLOCK DISTRIBUTION AND SCHEMA: FOR ALL COLLECTIVE CONSTRUCTION TASK ENVIRONMENTS. ENV: ENVIRONMENT.

Block Type Distribution and Construction Schema				
ENV	Type A Blocks	Type B Blocks	Type C Blocks	Construction Schema
1	5	5	0	A B A B A B A B A B
2	4	5	1	A B A B A B A B C B
3	3	5	2	A B A B A B C B C B
4	2	5	3	A B A B C B C B C B
5	1	5	4	A B C B C B C B C B
6	0	5	5	C B C B C B C B C B
7	0	6	4	C B C B C B C B B B
8	0	7	3	C B C B C B B B B B
9	0	8	2	C B C B B B B B B B
10	0	9	1	C B B B B B B B B B

TABLE II

CONSTRUCTION RULES: FOR BLOCK CONNECTION IN ALL COLLECTIVE CONSTRUCTION TASK ENVIRONMENTS.

Construction Rules			
Environment	A Connects:	B Connects:	B Connects:
[1, 10]	B	C	B
Robots Required	1	2	3

make all but the first connection. That is, only two robots are required to connect the first two blocks.

A. Experiment Design

Collective construction experiments measure the impact of the adaptation of robot team behavior by CONE-2 and NEAT on the number of *blocks connected* (team fitness) for a given simulation *environment*. An environment is defined as a distribution of block types, construction schema and rules. The objective was to ascertain the efficacy of CONE-2 versus NEAT for evolving specialized behaviors, where specialization enables cooperation and increases team fitness.

B. Fitness Evaluation

A robot's fitness is calculated based on the time taken (*T*) for it to move block *i* of a given type (BT_i) from an initial position in the environment, and to connect it to other blocks in the construction zone.

Block types *A*, *B*, and *C* yield different fitness values (table III) for being moved and connected in the construction zone. The varying fitness rewards for different block types reflect the degree of difficulty (cooperation) to connect given block types. Thus, a robot's fitness function (g_η) is the sum of V_η taken over all simulation iterations comprising the *lifetime* (table III) of robot η (equation 1).

$$g_\eta = \sum_t \sum_i V_{BT_i,\eta,t} \tag{1}$$

Where, $V_{BT_i,\eta,t}$ is the fitness gained by robot η after it connects block BT_i at simulation time *t*. If more than one robot *connects* block BT_i to other blocks in the construction zone, then equal fitness is given to each robot that cooperated to connect block BT_i.

G (equation 2) is team fitness, simply calculated as the sum of all fitness values for all *n* robots.

$$G = \sum_n g_\eta \tag{2}$$

The team's goal is to maximize *G*. However, robots do not maximize *G* directly, instead each robot η attempts to maximize its own fitness function (g_η), where g_η guides controller evolution. At the end of each robot's lifetime, g_η and *G* are normalized to the range: [0.0, 1.0].

C. Simulation

An experiment applies CONE-2 or NEAT to evolve team behavior for 500 generations. A generation comprises three *epochs*. One epoch is 3000 simulation iterations, representing a task scenario that tests different robot starting positions, orientations, and block locations in an environment.

CONE-2 begins with one genotype population, from which one robot controller is evolved. However, CONE-2 increments the number of populations (controllers), up to a team of 10 robots. NEAT evolves a fixed team size of 10 robots, where the current fittest controller is used for each robot in the team. Given the adaptation of team size by CONE-2 and controller topology by NEAT, both CONE-2 and NEAT experiments use the same number of genotype evaluations (for each experiment) to ensure a fair comparison. The fitness of CONE-2 and NEAT teams is an average calculated over 20 simulation runs of a given experiment. Table III presents the simulation, CONE-2 and NEAT parameter settings. These parameter values were determined experimentally. Minor changes to these values produced similar results for both CONE-2 and NEAT evolved teams.

D. Behavioral Specialization

The *degree of behavioral specialization* (*S*) exhibited by a controller is defined by the frequency with which the controller switches between executing distinct actions during its lifetime. The *S* metric used is an extension of that defined by Gautrais *et al.* [7], and was selected given its success is previous work [14], [15]. Equation 3 specifies the calculation of *S*, which is the frequency with which a controller switches between each of its actions during its lifetime. In equation 3, *A* is the number of times the controller switches between different actions, and *N* is the total number of possible action switches. Equation 3 assumes at least two controller actions.

$$S = \frac{A}{N} \tag{3}$$

An *S* value close to zero indicates a high degree of specialization, where a controller specializes to primarily one action, and switches between this and other actions with a low frequency. An *S* value close to one indicates a low degree of specialization, where a controller switches between some or all of its actions with a high frequency. A perfect specialist (*S* = 0), is a controller that executes the same action for the duration of its lifetime ($A = 0$). An example of a non-specialist ($S = 0.5$) is where a controller spends half of

TABLE III

SIMULATION AND NEURO-EVOLUTION PARAMETERS: FOR THE COLLECTIVE CONSTRUCTION TASK.

Simulation and Neuro-Evolution Parameters	
Simulation runs	20 (CONE-2 / NEAT)
Initial number of robots (Genotype populations)	1 (CONE-2 / NEAT)
Block / Robot detection sensor range	0.04
Robot size (diameter)	0.02
Maximum robot grip range	0.003
Maximum robot movement (d_{max})	0.04
Behavioral specialization threshold	0.5
Initial robot positions	Random (Excluding construction zone)
Simulation environment	Continuous
Environment width / height	1.0
Construction zone size (Diameter)	0.08
Block size (Width / Height)	0.01 (Type A / B / C)
Block Type A / B / C Fitness reward	2 / 5 / 10
Generations / Epochs	500 / 3 (CONE-2 / NEAT)
Iterations per epoch (Robot lifetime)	3000
Mutation (per gene) probability / Mutation range	0.05 / [-1.0, +1.0]
Population (sub-population) elite portion	20%
Weight (gene) range	[-1.0, +1.0]
Genotype length (Connection weights)	25 (CONE-2) / Variable (NEAT)
Genotypes per population	200 (CONE-2) / 600 (NEAT)
Initial number of populations	1 (CONE-2 / NEAT)
Maximum number of Populations (Robots)	10 (CONE-2)
Fitness Stagnation Period (Z generations)	5 (CONE-2)
Genotypes per sub-population	20 (CONE-2)
Sub-Populations (Hidden layer neurons)	10 (CONE-2)
Sensory Input Nodes	22 (CONE-2 / Maximum for NEAT)
Motor Output Nodes	3 (CONE-2 / Maximum for NEAT)
Initial Sensory Input Nodes	4 (NEAT)
Initial Motor Output Nodes	3 (NEAT)
Survival threshold	20% (NEAT)
Add node mutation probability	20% (NEAT)
Add sensor mutation probability	20% (NEAT)
Disjoint Coefficient	2.0 (NEAT)
Excess Coefficient	2.0 (NEAT)
Weight Difference Coefficient	1.0 (NEAT)
Compatibility Threshold	6.0 (NEAT)
Compatibility Modifier	0.3 (NEAT)
Interspecies mutation rate	5% (NEAT)
Mate multi-point probability	20% (NEAT)
Recurrent link probability	5% (NEAT)
Drop off age	15 generations (NEAT)

its lifetime switching between two actions. For example, if $A = 3$, $N = 6$, the controller switches between each of its actions every second iteration. Controllers are labeled as *specialized* if *S* is less than a given *behavioral specialization threshold* (for this study a 0.5 threshold was selected, table III). Otherwise, controllers are labeled as *non-specialized*.

V. ROBOT CONTROLLER

CONE-2 and NEAT evolved robots use the following sensory inputs and motor outputs in robot controllers.

A. Detection Sensors

Using CONE-2, a robot has 12 *block detection* ([SI-0, SI-11] in figure 2), three *block demand* ([S-12, S-14]), and four *robot detection* ([S-15, SI-16, S-17] in figure 2) sensors. A robot's sensory field of view is split into north, south, east and west sensor quadrants (SQ-0, SQ-1, SQ-2, and SQ-3, respectively, in figure 3). NEAT evolves differing combinations and numbers of sensor types (section V-E). All sensor values are normalized to the range [0.0, 1.0].

1) Block Detection Sensors: Block detection sensors are constantly active for the duration of a robot's lifetime. Sensor *q* returns the closest block type in quadrant *q*, divided by the squared distance to *this* robot.

2) Robot Detection Sensors: Robot detection sensors prevent collisions between robots and enable cooperation. Sensor *q* returns the closest robot in sensor quadrant *q*, divided by the squared distance to *this* robot.

3) Block Demand Sensors: These sensors are constantly active. At each simulation iteration the construction zone broadcasts a signal that is received by each robot's block demand sensors. This signal indicates the block type with the highest demand (next required in the construction sequence) at a given iteration. A sensor value of 1.0 indicates the

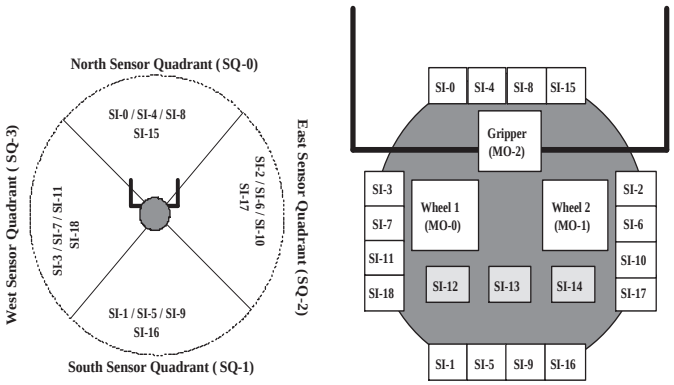

Fig. 3. *Left:* Robot sensor quadrants: SQ-1, SQ-2, SQ-3, SQ-4. Block demand sensors: SI-12, SI-13, SI-14 are not depicted since they work in all four quadrants. *Right:* Example sensory-motor setup. *SQ*: Sensor quadrant.

highest demand, 0.5 indicates second highest demand, and 0.0 indicates no demand. For example, block demand sensor S-12 will receive a demand signal equal to 1.0 if block type *A* is the next type required in the construction sequence. Sensor S-13 will receive a demand signal value of 0.5 if a type *B* block is the next type required after block type *A*. Sensor S-14 will receive a signal of 0.0 if a type *C* block is no longer required. If block types *B* and *C* are both required after type *A* (for example, where the type *B* and *C* blocks can be connected to either side of the type *A* block), then sensors S-13 and S-14 will each receive a 0.5 signal value.

B. Movement Actuators:

Two wheel motors control a robot's heading at constant speed. Wheel motors (MO-0 and MO-1 in figure 2) need to be explicitly activated. Movement is calculated in terms of real valued vectors (*dx* and *dy*). A robot's heading is determined by normalizing and scaling its motor output values by the maximum distance a robot can traverse [1] in one iteration (d_{max} in table III). That is:

$$dx = d_{max}(o_1 - 0.5)$$
$$dy = d_{max}(o_2 - 0.5)$$

Where, o_1 and o_2 are MO-0 and MO-1 output values, respectively. To calculate the distance between this robot, other robots and blocks in the environment, the squared Euclidean norm, bounded by a minimum observation distance is used.

C. Block Gripper:

Each robot is equipped with a gripper to transport blocks to the construction zone. The gripper is explicitly activated. If no block is held, the robot grips the closest block within gripper range (table III). If the robot is gripping a block, then the block is released. Gripper output is normalized in the range [0.0, 1.0]. For the collective construction task, gripping different block types are considered different actions for controller specialization (section IV-D).

TABLE IV

CONE-2 EVOLVED TEAMS: BEHAVIORAL COMPOSITION. ROBOTS IN A TEAM WERE CALCULATED AS NON-SPECIALIZED, OR SPECIALIZED TO GRIPPING TYPE A, B, OR C BLOCKS. NEAT IS NOT INCLUDED SINCE THE FITTEST NEAT EVOLVED TEAMS WERE BEHAVIORALLY HOMOGENOUS (NON-SPECIALIZED) FOR ALL ENVIRONMENTS.

CONE-2 Evolved Teams: Behavioral Composition				
Environment	A Block Specialist	B Block Specialist	C Block Specialist	Non-Specialist
1	0	0	0	3
2	4	4	2	0
3	4	4	2	0
4	3	4	3	0
5	1	4	5	0
6	0	5	5	0
7	0	5	5	0
8	0	6	4	0
9	0	6	4	0
10	0	8	2	0

D. ANN Controller (CONE-2)

CONE-2 evolved robots use a recurrent ANN controller [5], fully connecting 22 sensory input neurons to 10 hidden layer neurons to three motor output neurons (figure 2, left). Hidden and output neurons are sigmoidal [10] units. Sensory input neurons [SI-19, SI-21] have recurrent connections that accept the previous activation state of the output layer. At each simulation iteration of the robot's lifetime the motor output with the highest value is the action executed.

1) *MO-0, MO-1:* Calculate direction of movement from motor outputs MO-0 (*dx*) and MO-1 (*dy*).
2) *MO-2:* Activate gripper.

E. NEAT: Initial ANN Controller

NEAT evolved robots begin with a simple ANN controller (figure 2, right, presents an example), that is subject to *complexification* during NEAT adaptation. To ensure that NEAT controllers execute actions and accomplish the collective construction task with some degree of success, NEAT controller motor outputs are kept the same as the CONE-2 controller (figure 2, left). The example initial controller presented in figure 2 (right) uses four sensory input neurons fully connected to motor outputs. In this example, sensor inputs are, one *block type A detection sensor* (SI-0) using the robot's north sensor quadrant (SQ-0), one *block type A demand sensor* (SI-1) using all four sensor quadrants ([SQ-0, SQ-3]), one robot detection sensor (SI-2) using the robot's north sensor quadrant (SQ-0), and one bias node (SI-3). The bias node uses a constant weight value of 1.0. However, NEAT controllers are randomly initialized with three sensor inputs and a bias node connected to motor outputs.

VI. RESULTS AND DISCUSSION

Figure 4 presents the average *team fitness* (section IV-B) for CONE-2 and NEAT teams evolved in each environment. Statistical comparisons (independent t-tests) indicate that CONE-2 evolved teams yield a significantly higher average team fitness, compared to NEAT evolved teams,

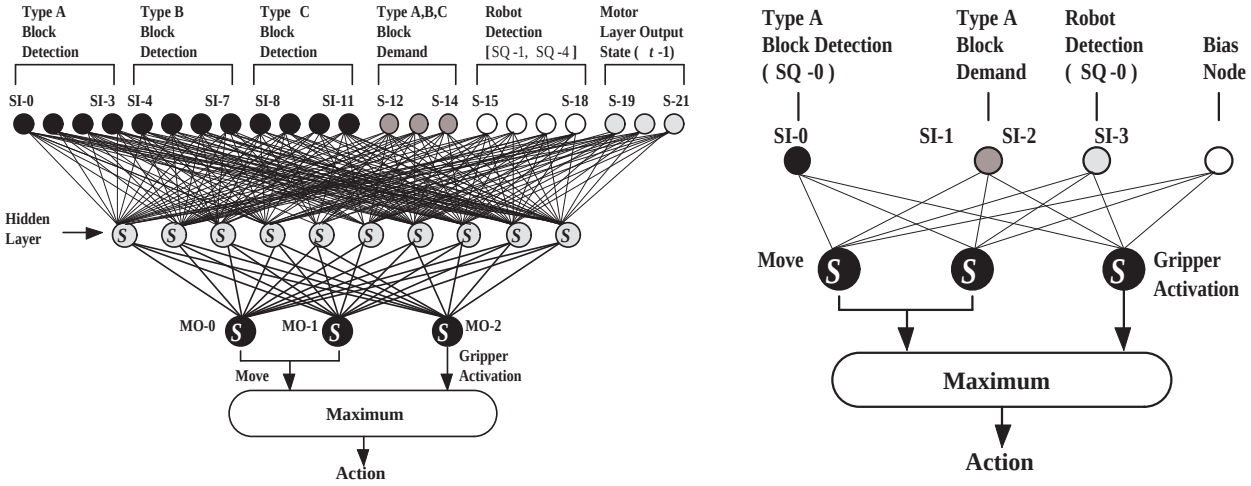

Fig. 2. *CONE-2 (Left) NEAT (Right) Controllers.* NEAT initializes a homogenous controller population subject to *complexification*. The NEAT controller illustrated is an example only. Controllers are initially random sensory-motor configurations (right). CONE-2 uses a controller with a fixed topology (left).

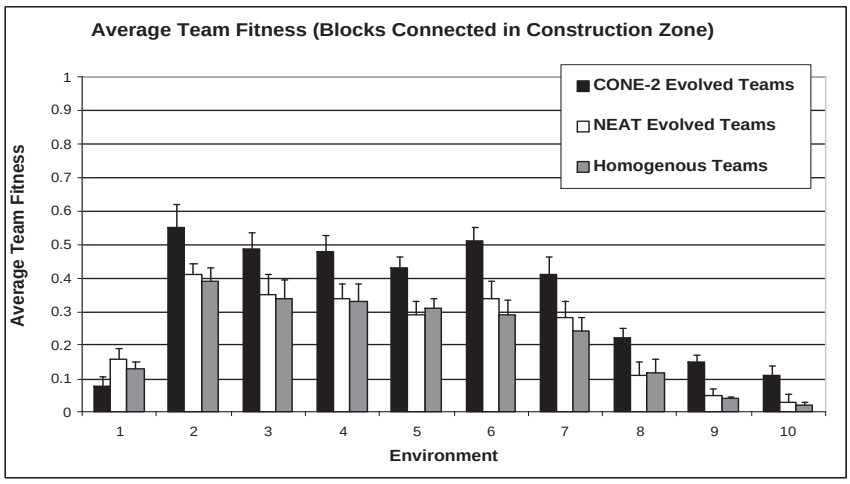

Fig. 4. *Average Team Fitness:* CONE-2, NEAT and Homogenous teams (the latter establish behavioral heterogeneity benefits. See text for details).

for environments [2, 10]. Thus, CONE-2 evolved teams achieved a higher average team fitness (compared to NEAT evolved teams), in environments requiring cooperation between at least two robots. These results partially support this study's hypothesis (section I-A). In environment 1, where no cooperation was required for task accomplishment, NEAT evolved teams yielded a higher average task performance (with statistical significance). In figure 4, the average team fitness has been normalized to the range: [0.0, 1.0]. This was done in order that team fitness's be readily comparable as portions of maximum task performance.

Table IV presents the behavioral composition of the fittest teams, evolved by CONE-2, in each environment. Each of the fittest CONE-2 evolved teams, except that evolved in environment 1, were *behaviorally heterogeneous*. In this study, behavioral heterogeneity is when a team contains at least two behavioral specializations. Behavioral specializations were calculated based on the time for which robots gripped type *A*, *B* and *C* blocks during their lifetimes. That is, a robot was defined as specialized to gripping block type *x*, if the robot switched between gripping block type *x* and other block types with low frequency during its lifetime (section IV-D). Movement was not considered by the specialization metric since a robot's move action did not directly contribute to block construction. Table IV does not include the behavioral composition of the fittest NEAT evolved teams, since these teams were *behaviorally homogenous* for all environments. That is, all robots in the fittest NEAT evolved teams adopted the same (non-specialized) behavior.

These results indicate the higher the degree of cooperation required for task accomplishment (one versus two or three robots to connect given block types), the more behavioral heterogeneity in CONE-2 evolved teams. That is, depending on the degree of cooperation required to connect blocks, CONE-2 evolved teams that contained varying complements of behavioral specializations. Such behavioral heterogeneity enabled cooperation and the comparatively high average team fitness of CONE-2 evolved teams.

Table IV also indicates that for all environments (except environment 1), CONE-2 evolved the maximum team size of 10 robots. For environment 1, CONE-2 evolved a 3 robot team, since no cooperation was required to connect blocks (tables I and II). After CONE-2 had increased the team size to three robots, there was no further fitness stagnation, and thus no incentive for adding populations (controllers).

Figure 4 also presents results from re-running the fittest CONE-2 evolved teams as *behaviorally homogenous* teams. This was done via randomly selecting one controller in the fittest CONE-2 evolved team (for each environment), and using this as the controller for all robots in a team. The same team sizes as evolved by CONE-2, for each environment, were used. These *homogenous teams* were run in all environments, for 20 runs of one *robot lifetime* (table III), and an average team fitness calculated. These homogenous team experiments did not apply any form of adaptation. A statistical comparison indicated that the average team fitness of homogenous teams was comparable to NEAT evolved teams, but lower than CONE-2 evolved teams, for all environments. The exception was environment 1, where homogenous teams yielded an average team fitness comparable to CONE-2 evolved teams.

This result fully supports the hypothesis (section I-A) that CONE-2 is appropriate for evolving teams with behavioral heterogeneity which in turn leads to a higher average team fitness (compared to NEAT evolved teams) in environments where cooperation is required. However, for environment 1, CONE-2 evolved teams yielded a lower average fitness (with statistical significance), compared to NEAT evolved teams (figure 4). This was a result of NEAT using a fixed team size of 10 robots, and CONE-2 using only three robots. More robots in NEAT evolved teams, and the lack of any cooperation requirements in environment 1 resulted in a higher average task performance for NEAT evolved teams.

VII. Conclusions

This paper described a study that applied CONE-2 and NEAT as controller evolution methods in simulated robot teams that must solve a collective construction task. The study's goal was to elucidate the most appropriate method for encouraging behavioral specialization (during controller evolution), where such specialization enables cooperation. The collective construction task often required cooperation to connect blocks and to achieve the highest team fitness.

Results indicated that CONE-2 evolved teams achieved a higher average team fitness (compared to NEAT evolved teams) in 90% of environments tested. Team fitness was defined as the number of blocks connected together, in a construction zone, in the correct sequence. Also, these results supported a hypothesis that behavioral heterogeneity (complementary behavioral specializations) in teams facilitate cooperation, which in turn leads to higher average team fitness in this collective construction task. However, the casual mechanisms and relationships between behavioral heterogeneity, cooperation and collective behavior task performance is the subject of ongoing research.

References

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1–12, New York, USA, 2004. Springer.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, Oxford, England, 1998.

[3] D. D'Ambrosio and K. Stanley. Generative encoding for multiagent learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1–8, Atlanta, USA, 2008. ACM Press.

[4] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Germany, 2003.

[5] J. Elman. Finding structure in time. *Cognitive Science*, 14(1):179–211, 1990.

[6] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[7] J. Gautrais, G. Theraulaz, J. Deneubourg, and C. Anderson. Emergent polyethism as a consequence of increased colony size in insect societies. *Journal of Theoretical Biology*, 215(1):363–373, 2002.

[8] F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(1):317–342, 1997.

[9] D. Hawthorne. Genetic linkage of ecological specialization and reproductive isolation in pea aphids. *Nature*, 412(1):904–907, 2001.

[10] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, 1991.

[11] L. Li, A. Martinoli, and A. Yaser. Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior.*, 12(3):199–212, 2004.

[12] A. Murciano, J. Millan, and J. Zamora. Specialization in multi-agent systems through learning. *Biological Cybernetics*, 76(1):375–382, 1997.

[13] G. Nitschke. *Neuro-Evolution for Emergent Specialization in Collective Behavior Systems. PhD thesis*. Computer Science Department, Vrije Universiteit, Amsterdam, Netherlands, 2009.

[14] G. Nitschke, M. Schut, and A. Eiben. Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task. *Evolutionary Intelligence*, 3(1):13–29, 2010.

[15] G. Nitschke, M. Schut, and A. Eiben. Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm and Evolutionary Computation*, 2(1):25–38, 2012.

[16] M. Potter and K. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[17] M. Potter, L. Meeden, and A. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1337–1343, Seattle, USA, 2001. AAAI Press.

[18] M. Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, USA, 1997.

[19] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[20] C. Schultz and L. Parker. In *Multi-robot Systems: From Swarms to Intelligent Automata*. Kluwer, Washington DC, USA, 2002.

[21] K. Stanley. *Efficient Evolution of Neural Networks Through Complexification. Ph. D. Dissertation*. Department of Computer Sciences, The University of Texas, Austin, USA, 2004.

[22] K. Stanley, N. Kohl, R. Sherony, and R. Miikkulainen. Neuroevolution of an automobile crash warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Washington DC, USA, 2005. ACM Press.

[23] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation.*, 10(2):99–127, 2002.

[24] K. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100, 2004.

[25] K. Stanley and R. Miikkulainen. Evolving a roving eye for go. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Seattle, USA, 2004. ACM Press.

[26] X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(3):203–222, 1993.

[27] C. Yong and R. Miikkulainen. Coevolution of role-based cooperation in multi-agent systems. *IEEE Transactions on Autonomous Mental Development*, 1:170–186, 2010.